02

Introducción a polimorfismo

Transcripción

[00:00] Bien, siguiendo aquí con lo que dejamos en la clase anterior, vamos a profundizar un poco más en lo que es el concepto de herencia. Herencia tiene dos pilares. El primero es la reutilización del código. Ya vimos que gracias a la herencia podemos escribir una vez un solo método y reutilizarlo sobre todas las clases hijas.

[00:26] Y también tenemos el caso de polimorfismo. Polimorfismo es un concepto un poco más abstracto por así decirlo, pero no se preocupen, lo vamos a ver detalladamente aquí. Para esto vamos a crear aquí una clase, ya sabemos ese proceso, y la vamos a llamar TestReferencias, porque polimorfismo tiene mucho que ver con referencias.

[00:56] Ustedes recuerdan que en el curso pasado ya hemos trabajado mucho con lo que son las referencias a nivel de código. Recuerden que Java trabaja todo en base a referencias, esa es una premisa que tienen que recordar ahora. Y bueno, vamos a hacer lo que ya hemos hecho antes, vamos a crear nuestro método main, primero que todo. Listo.

[01:18] Y aquí dentro vamos a crear un funcionario igual new funcionario, espacio, perfecto, y a ese funcionario pues le vamos a decir Funcionario por ejemplo nombre, setNombre. Le vamos a poner Diego. Y ahora vamos a crear pues un gerente, entonces vamos a darle gerente gerente igual new Gerente, perfecto. Nos olvidamos aquí el punto y coma. No hay problema.

[02:07] Y aquí le vamos a asignar también otro nombre. Punto setNombre y a ese gerente vamos a ponerle de nombre Jimena, nombre de una mujer. Listo. Y a cada uno le vamos a asignar un salario, entonces funcionario.setSalario, el funcionario vamos a decir que gana 2000, y la gerenta, perdón, gerente.setSalario, ella gana 10000. Perfecto.

[02:50] Entonces hasta aquí no hemos hecho nada nuevo, de hecho estamos repitiendo mucho código de la clase anterior. ¿Cuál es el punto al que yo quiero llegar ahora? Si ustedes ven aquí, sabemos que esta clase de aquí, gerente, ¿hereda de quién? De funcionario. Recuerdan eso.

[03:16] Gerente tiene un extend de funcionario, por lo cual nosotros tenemos acceso a los campos de funcionario y a los métodos de funcionario, recuerdan. Entonces aquí en nuestro test de referencias funcionario podría ser igual entonces a un gerente, recordemos ahora lo que son referencias.

[03:38] Entonces aquí, sabiendo que gerente hereda de funcionario, yo también podría decir algo así como que este funcionario es un nuevo gerente y vemos que el código compila. Ustedes se pueden estar preguntando ¿pero cómo es posible? Aquí está el detalle. Como gerente hereda de funcionario, entonces podemos nosotros decir que todo gerente es un funcionario.

[04:10] Para que puedan hacer bien la analogía, vamos a hacer un pequeño comentario aquí. La clase más general, por así decirlo, el elemento más genérico, puede ser adaptado al elemento más específico. Vamos a continuarlo aquí. Al elemento más específico. ¿Cuál es el elemento más genérico? Funcionario. ¿Por qué? Porque es de él de quien están extendiendo.

[04:51] ¿Y quién es el más específico? Gerente. Ahora, tenemos acá nuestra primera analogía, piensen en esto. ¿Será que podemos invertir la selección, por ejemplo decir que este gerente es un nuevo funcionario? Vamos a probar. Funcionario, no compila. ¿Por qué no está compilando? Acá hay un detalle que tienen que tener en cuenta, muy presente ahí.

[05:22] ¿Este código no está compilando por qué? Porque no todos los funcionarios son gerentes, entonces ven cómo va el orden de la analogía. En el caso de aquí es todos los gerentes son funcionarios. ¿Por qué? Porque funcionario es el elemento más genérico. En el caso de gerente no, porque no todos los funcionarios son gerentes. Gerente es un tipo de funcionario, es un hijo de la clase funcionario.

[05:50] Entonces esta analogía no va a funcionar, no va a compilar aquí. Por lo cual acá si yo deseo de verdad crear un nuevo gerente, entonces yo tengo que crear un nuevo gerente explícitamente. Y ahora, nosotros habíamos especificado ya ciertas diferencias entre un gerente y un funcionario. ¿Cuál era la diferencia del gerente?

[06:17] Aparte de la bonificación el gerente tenía dos métodos nuevos que eran iniciar sesión y setear la clave para iniciar la sesión. ¿Qué pasaría si yo deseo aquí hacer que mi funcionario, que está inicializado como nuevo gerente, inicie sesión? Bajo aquí y le digo funcionario.iniciarSesion y vemos que él no está compilando, de hecho él no está encontrando el método.

[06:53] Voy a verificar que el método se llame iniciarSesion y no iniciaSesion porque creo que está mal. Es iniciarSesion, incluso lo voy a copiar y aquí lo voy a pegar. Solamente para saber que estoy llamando al método correcto. Aún así él me está diciendo aquí ¿sabes qué? El método iniciarSesion en la clase funcionario no existe.

[07:15] Y ahora las cosas ya se comienzan digamos a poner un poco más oscuras. ¿Por qué? Porque se supone que yo estoy acá creando un nuevo gerente. ¿Entonces por qué él no tiene el método, si se supone que es un nuevo gerente? No entiendo. ¿Esto del polimorfismo cómo es? Tranquilo. En realidad la explicación para esto es muy simple, y vuelvo al inicio.

[07:39] ¿Recuerdan que en Java todo son referencias? Si yo estoy tomando esta referencia de funcionario, entonces en funcionario, vamos al código, ¿será que

yo tengo ese método iniciar sesión? No, porque es un método propio del gerente. Entonces, aquí, en este caso, yo necesitaría crear un nuevo gerente para que en el caso de Jimena por ejemplo, si yo pusiera gerente, entonces él sí me va a permitir escribir este método. ¿Por qué?

[08:12] Porque acá me dice: "el método de iniciarSesion necesita un documento string". Si yo le pongo cualquier cosa aquí por ejemplo, hizo punto y coma al final, vemos que el compila tranquilamente. ¿Por qué? Porque gerente, la referencia de gerente, el objeto de acá, la referencia sí tiene ese método de iniciarSesion. Entonces, repasando un poco, funcionario no incluye el método iniciarSesion porque la referncia de funcionario no incluye ese método.

[08:50] Funcionario puede estar inicializado como un nuevo gerente? Sí, sí puede, porque todos los gerentes son funcionarios. Entonces el tipo más genérico sí es posible inicializarlo como un tipo más específico. Pero la referencia del objeto va a seguir siendo la del tipo genérico. Perfecto.

[09:09] En el caso de inicialización ya es conocida, y bueno, acá sí vamos a tener todos los métodos propios de esta clase. Quizás ahora se están preguntando: "Entonces esto de la herencia, perdón, esto del polimorfismo, ¿para qué sirve?" porque hasta este punto no tiene utilidad. Y es verdad, no tiene utilidad.

[09:33] Esto solamente fue digamos un ejemplo de qué es el polimorfismo. En el siguiente ejemplo ya vamos a ver una aplicación de este concepto y les aseguro que la van a usar muy, muy seguido en su trabajo el día a día.